NFTwrapper V01 Audit Report Notes

Audit by Armen Arakelian

The audit was thorough and revealed several key issues that have been addressed and fixed. Most of the issues were related to depositing NFTs within an ERC-1155 capable vault. Those were related to possible integer overflows. Checks have been added to the relevant areas to ensure the math is safe. There were also several suggestions to improve gas optimization that were incorporated. However, there is an inaccuracy within the Appendix that must be clarified.

The NFT ID limit of >2^60 is a soft limit. NFTs with a greater ID can be stored, however the combination of NFT ID + contract index has the potential for a collision. This is not an issue if the vault only allows a single contract, as the only possible collisions are any the underlying data structure of a mapping might introduce. The collision risk does become an issue for multi-contract vaults, but there are some caveats. In all cases, NFTs will still be able to be withdrawn. The worst-case scenario of a collision is when one of the two (or more) NFTs is withdrawn. The other NFT(s) will return a false negative when calling `getTokenIndex()`.

If the index of the NFT within the vault is known (verifiable by calling `tokenAt()`), you can still withdraw that NFT by calling `withdrawTokenAtIndex()`. This does carry a slight risk of withdrawing an undesired NFT if the index of the FNT changes (due to how withdrawals modify the order of NFTs within a vault). This will only come into effect if the desired NFT is (or becomes) the last NFT in the vault and a withdrawal transaction is confirmed earlier than the user's transaction. In this case, as the vault uses the "pop and swap" method of withdrawing, the NFT index will change. Again, the NFT is still withdrawable, however the new index must be used to withdraw the NFT.

Withdrawing by NFT ID is only necessary if: the vault is a heterogenous vault, two NFTs from separate contracts are deposited whose combination of contract index and NFT ID are identical, and one of those NFTs is withdrawn. There is an assumption that NFT IDs will be minted sequentially. In this case, due to the technological limitations of minting and tracking >2^60 separate NFTs, this risk is essentially zero. For non-sequentially minted NFTs, the risk is dependent on the specific use case and interactions of the vault.

Jonathan Hineline | CEO, Bittrees
jhineline@bittrees.io

Ben Goetz | CTO, Bittrees
bgoetz@bittrees.io

Vault721ERM_V01.sol

[1] `_withdrawToken()` [HIGH] [FIXED]

Fixed the issue where the count of tracked NFTs is increased instead of decreased.

[2] `setRoles()` [HIGH] [FIXED]

Fixed the issue where roles could be unintentionally set and simplified the function to overwrite pre-existing roles.

[3] `changeContractAddress()` [MEDIUM] [REMOVED]

This function has been removed as functionality is redundant with `removeContractAddress()` and `addContractAddress()`.

[4] `_depositor` [LOW] [NO CHANGE NEEDED]

This has to be wiped each time to ensure that the reentrancy guard in `onERC721Received()` doesn't incorrectly allow an NFT to be stored. If it is not wiped, then anyone can send an NFT if the sender matches the previous depositor.

[5] `canDepositToken()` [LOW] [FIXED]

The redundant storage read was removed as per the suggestion.

[6] `withdrawTokenAtIndex()`, `withdrawTokens()`, `_withdrawToken()` [LOW] [FIXED]

The redundant storage reads were removed as per the suggestion.

Vault1155ERM_V01.sol

[1] `setRoles()` [HIGH] [FIXED]

Fixed the issue where roles could be unintentionally set and simplified the function to overwrite pre-existing roles.

[2] `deposit()` [HIGH] [FIXED]

Integer overflow issues have been resolved using overflow assertions. Additional possible overflows were fixed in `withdraw()` and `withdrawTokenAtIndex()`.

[3] `changeContractAddress()` [MEDIUM] [REMOVED]

This function has been removed as functionality is redundant with `removeContractAddress()` and `addContractAddress()`.

[4] `canDepositToken()` [LOW] [FIXED]

The redundant storage read was removed as per the suggestion.

[5] `withdrawTokenAtIndex()`, `withdrawTokens()`, `_withdrawToken()` [LOW] [FIXED]

The redundant storage reads were removed as per the suggestion.